Welcome to the Superpixel Benchmark Toolbox

If there are any questions, feel free to contact me: peer.neubert@etit.tu-chemnitz.de

Overview

This toolbox is intended to allow for comparable evaluation of superpixel segmentation algorithms. It provides an interface to datasets, implementation of the error metrics and methods for evaluation.
To evaluate a new algorithm, one has to:

1. Decide which metrics are required.
2. Download the toolbox, its dependencies and the required dataset for the chosen metrics.
3. Write an configuration file for your algorithm (YAML).
4. Call the benchmark with this configuration file (and be patient).

The toolbox requires Matlab and was only tested using Linux. It combines and extends our two previously published benchmark toolboxes [2] and [3]. Comparison of 14 available superpixel segmentation algorithms using this toolbox can be found in [1].

The numerical results (figures and .mat files) are provided in the numerical_results subfolder.

Details on the metrics and references

The following table gives an overview of the provided metrics and the used datasets:

Criteria	Metric	Required data	Used dataset
Runtime		Arbitrary images	BSDS500 test
Quality	Boundary recall	Manually annotated figure-ground segmentations	BSDS500 test
	Undersegmentation Error	Manually annotated figure-ground segmentations	BSDS500 test
	MASA	Manually annotated figure-ground segmentations	BSDS500 test
Stability	Stability to affine image transformations measured by Precision-Recall	Arbitrary images, transforms and transformed images	BSDS500 test
	Stability to image noise measured by Precision-Recall	Arbitrary images, noise models	BSDS500 test
	Motion Undersegmentation Error	Image sequences with ground truth optical flow	Sintel, KITTI
Compactness	Geometric properties: Standard deviation of size; isodiametric and isoperimetric quotients	Arbitrary images	BSDS500 test
	Motion Discontinuity Error	Image sequences with ground truth optical flow	Sintel, KITTI

For details on the error metrics and evaluation results have a look at the following papers:

- [1] Complete toolbox with most recent and comprehensive descriptions and results:
  - Neubert, P. (2015). Superpixels and their Application for Visual Place Recognition in Changing Environments. Dissertation, TU Chemnitz.
- [2] Metrics based on ground truth optical flow (using Sintel and KITTI datasets):
  - Neubert,P., Protzel, P. (2013). Evaluating Superpixels in Video: Metrics Beyond Figure-Ground Segmentation. *Proc. of British Machine Vision Conference (BMVC)*, Bristol, England
- [3] Benchmarks based on human manual ground truth labels and affine transformations (using BSDS data):
  - Neubert, P., Protzel, P. (2012). Superpixel Benchmark and Comparison. *Proc. of Forum Bildverarbeitung*, Regensburg, Germany

Please refer to one of these papers if you use this toolbox in your own work.

Getting Started Guide

Prerequisites

This toolbox depends on the BSDA, Sintel, and Kitti datasets, and a Yaml toolbox. Before you can start, you have to download them and set some paths.

1. Download and configure BSDS, Sintel and/or KITTI datsets

- BSDS http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html#bsds500
- Sintel http://sintel.is.tue.mpg.de/
- KITTI http://www.cvlibs.net/datasets/kitti/

There are YAML configuration files for each dataset in the *configs* folder (e.g. BSDS500.yaml). There, the paths to the datasets have to be configured. Although the affine transformations and noise benchmarks also use the BSDS 500 data, there are individual configuration files, where the path has to be configured.
To configure Sintel and KITTI, the variables SINTEL_PATH and KITTI_PATH in the *set_paths.m* have to be adapted.

2. Download Yaml toolbox and extract the Zip file

- http://yamlmatlab.googlecode.com/ (we use version 0.4.3)

The resulting location has to be placed on the Matlab path by configuring the *set_paths.m*

3. Set paths by calling *set_paths.m*

How to benchmark an Algorithm

The run of each benchmark metric typically consists of three steps:

1. Segment the required images
2. Compute the metric
3. Evaluate the results

In this toolbox, segmentation algorithms and datasets are configured using YAML files and functions are provided that use these configuration files. E.g., to evaluate the undersegmentation error and boundary recall, we require the BSDS dataset configuration file *configs/bsds500.yaml* and have to call the following three toolbox functions:

1. main_segmentBSDS(alg_paramfilename, '*configs/bsds500.yaml*');
2. main_runBenchmarkBSDS(alg_paramfilename, '*configs/bsds500.yaml*');
3. main_evaluateBSDS(alg_paramfilename, '*configs/bsds500.yaml*');

All intermediate results are stored on disk at the *results* folder.

You can use the provided example "segmentation" algorithm *segment_box.m* and its example config in *configs/box.yaml* or create a configuration for your own algorithm. The box algorithm simply divides the image in a regular grid.

Have a look at the Matlab script *main_benchmarkAlgorithm.m*. This script calls functions to segment all images, compute error metrics and visualize the results. Calling this script directly will only work if you have all datasets and will take some time (several hours!).

Feel free to remove some metrics in *main_benchmarkAlgorithm*.m if you use just a single dataset or want to compute only one of the error metrics.

For quick tests, you can also reduce the amount of used data in the dataset configuration files. Make sure to use the full dataset when comparing to our results (this is the provided configuration).

For the Sintel dataset, there are a lot of intermediate results stored to your hard disk drive, make sure to have enough free space (~30 GB).

The configuration of an algorithms requires the following fields (see *config/box.yaml* for an example):

id … identifier of the algorithm
segSaveDir … where to store the results
segParams … configuration of the segmentation
  path … path to add to the Matlab path
  segFct … A string that contains the function call of your segmentation algorithm. It is required to compute the Label image L and time t from an RGB input image I. Other parameters have to be given in the form params.segParams.set{s}.PARAMETERNAME, where s is the index of the parameterset and PARAMETERNAME is the name of the parameter. There can be multiple parameters with different names.
  set ... These are the parameter configurations. There has to be a name for each parameter set and a value for each parameter used in the segFct.
  oneShotSetName … for some metrics, just a single segmentation is evaluated. This should create about 250 segments. Place the name of a suitable parameterset here.

Precomputed Noise images

To ensure comparability of the robustness evaluation with respect to image noise, there are precomputed noise images available. They have to be placed (or linked) in a data folder in the toolbox directory.

- https://www.tu-chemnitz.de/etit/proaut/forschung/cv/segmentation.html.en#Superpixel_Benchmark